

# 《Java 编程技术》

## 实验报告










专业： 物理学（师）  
学号： 2014210967  
姓名： 官展鹏

## 1. 总体设计

这是一款扫雷游戏，界面设计和功能完全模仿 win7 的扫雷，每一个雷的每一个状态的显示图片都是从 win7 上截屏后 Ps 编辑得到的，实际上最初记录了许多 win7 扫雷的功能和在各种状态下点击后的反应，比如保存游戏的功能，各种模式下排名取前 5 的功能，但最后由于时间关系，只能实现部分功能（win7 版扫雷记录见附件“扫雷.docx”），如果会玩扫雷的话，真诚建议先点开 Demine.jar 文件玩一玩再看这篇实验报告。

### 游戏界面：

扫雷分成“初级”、“中级”、“高级”三个等级，分别对应（9×9 10 个雷|16×16 40 个雷|16×30 99 个雷），雷的分布在鼠标第一次点击之后才开始，分布好雷后再设置每一个非雷位置的数字，目的是为了防止第一次就点到雷（实际上扫雷是先设置好雷，如果第一次点击到雷，就把雷移动到左上角，如果左上角有雷，就移动到邻近的未知），对应每一个扫雷按钮有如下几种状态和点击反应：







- 1)  和  左击则点开，对所在位置的八个方向进行宽度优先搜索，直到碰到数字，则停止搜索，被搜索到的所有方块都被翻开，变为 3) 中的一种，如果是雷，则变为 ，游戏失败，停止计时。右击则标记为 。双击，如果是游戏未开始，则双击被视为点开，若游戏已经开始，双击则附近的  都会被提示为 。
- 2)  表示该点被标记为雷，左击无反应，右击则变为 ，双击反应同 1)
- 3)  表示被点开的安全方块，方块上的数字表示该方块附近八个方块中，含有的雷的数量，左击、右击都无反应，双击根据周围八个被标记为  的个数  $n$ ，如果  $n$  与数字相同，则八个方块中其余的未被点开的一次被点开，反应与 1) 被左击相同。如果  $n$  不等于数字，则未被点开的  都被提示为 。
- 4)  如果游戏失败，则被点到的雷变为第一张图，的没有被点又没有被标记的雷，变为第二张图，被标记正确的雷变为第三张图。

### 游戏结束：

把所有非雷方块点开即胜利，如果完成时间超过对应难度下的当前记录，则弹出窗口让用户输入名字，否则弹出恭喜窗口。  
排雷过程中点到雷则失败。

## 类的设计:





## 主要的类:

- >  Demine
- >  Difficult
- >  Direction
- >  Main
- >  Rank
- >  Status

## 枚举型 Status:

包括如下常量与对应值（图片太长就不截常量了）：

```
NONE(0), ONE(1), TWO(2), THREE(3), FOUR(4), FIVE(5), SIX(6), SEVEN(7),  
EIGHT(8), BUTTON(9), FLAG(10), BOOM(11), DOUBT(12), FAILED(13),  
FAILFLAG(14), PROMPT(15);
```

- ▼  Status
  -  getStatus(int) : Status
    - value
  -  Status(int)
  -  getValue() : int

**Status**类用来表示扫雷中每一个方块的状态，每一种状态与对应显示的图片名相同，因此可以直接调用默认**toString()**方法获取对应状态的图片，该类拥有私有构造函数，将每个常量与括号中的值对应起来，主要是为了关联非雷方块附近的雷的数量计数与该方块需要显示的数字图片。该枚举型的函数成员**getStatus(int)**方法可直接由数字生成对应状态，**getValue()**方法可获得状态对应的数字。

### Difficult 类:

**Difficult**类用于记录在不同难度下的类的行列值与雷数目，只包含如下值：

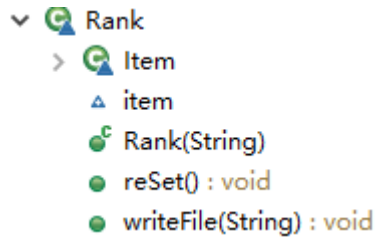
```
static final int[][] Diff = {{9, 9, 10}, {16, 16, 40}, {16, 30, 99}};
```

## Direction 类:

用于记录上下左右八个方向对应的二维坐标运算，方便宽度优先搜索时的循环判断，只包含如下值：

[illegible]

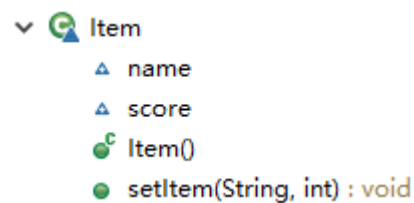
## Rank 类:



用来记录排行榜数据。

## Rank 内部类:

### Item 类:



用于表示单个记录的用户名与成绩。

### Item 数据成员:

**name:** 用户名  
**score:** 最好成绩

### Item 函数成员:

**Item():** score 默认为 999, name 默认为“匿名”。  
**setItem(String, int):** 设置对应数据成员。

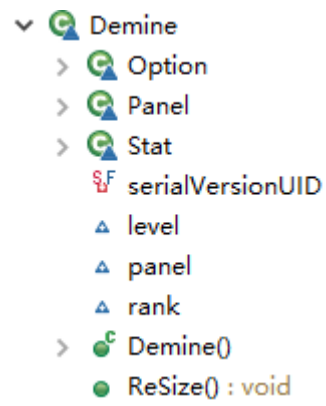
## Rank 数据成员:

**item[3]:** 是内部类 Item 的实例化, 用来记录在不同难度下的最好成绩与对应用户名, 数组大小始终为 3。

## Rank 函数成员:

**Rank(String filename):** 从文件中读取排行榜, 如果文件不存在, 则新建文件, 写入三种难度下的默认 item 值。  
**reSet():** 重置 item 数组的变量为默认值。  
**writeFile(String filename):** 将三种难度的 item 值写入文件。

## Demine 类:



```
graph TD
    Demine --> Option
    Demine --> Panel
    Demine --> Stat
    Demine --> serialVersionUID
    Demine --> level
    Demine --> panel
    Demine --> rank
    Demine --> DemineMethod[Demine()]
    Demine --> ReSizeMethod[ReSize() : void]
```



Demine

- > Option
- > Panel
- > Stat
- serialVersionUID
- level
- panel
- rank
- > Demine()
- ReSize() : void

Demine 类继承自 JFrame 类，是游戏的整个窗口。

## Demine 内部类:

### Option 类:

```
▼  Option
    serialVersionUID
    ▲ select
    >  Option(JFrame, int)
```

Option 类继承自 Jdialog 类，负责选择游戏难度。



### Option 数据成员:

**select:** 用于记录被选择的难度，由三个 JRadioButton 的监听器负责修改，只有确认键被按下时，才会被设置成游戏难度。

### Option 函数成员:

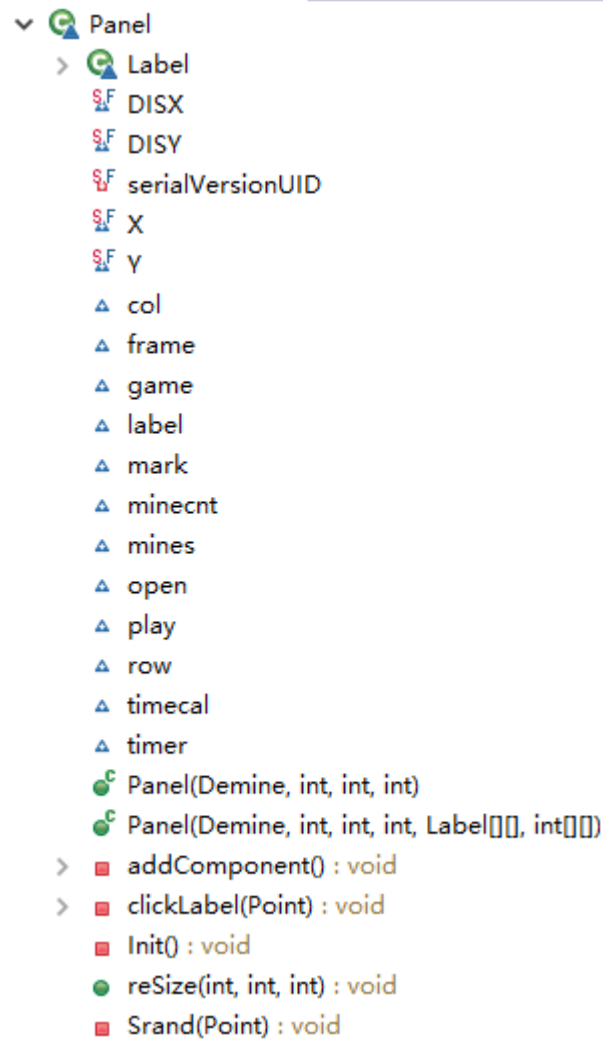
**Option(Jframe, int):** 传入参数为主面板 frame，数字表示当前难度，由易到难分别为 0，1，2

### Stat 类:

```
▼  Stat
    serialVersionUID
    >  Stat(JFrame)
```

Stat 类继承自 Jdialog 类，用于显示排行榜，可在排行榜中重置各个难度的最高记录。

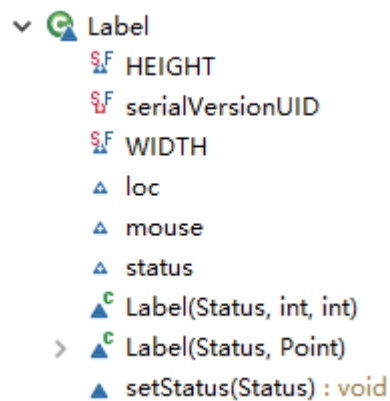
## Panel 类:



`Panel` 类继承自 `JPanel` 类，负责扫雷游戏的主界面，主要包括  $m \times n$  个二维方块、计时器、旗子个数。

## Panel 内部类:

### Label 类:



Label 类继承自 JLabel 类,用来显示每一个小方块并记录其状态。

### Label 静态常量:

**HEIGHT:** 表示每一个方块的高。

**WIDTH:** 表示每一个方块的宽。

### Label 数据成员:

**loc:** Point 类对象,用来记录方块所在的行和列的值,方便宽度优先搜索时获取坐标。

**mouse:** 用于记录鼠标点击状态,由 Label 的鼠标监听器负责改变,Press 鼠标则加上鼠标的 getButton()值,Release 鼠标则减去鼠标的 getButton()值,当其等于 4 时立即置零(左右键同时点击状态),这样在 Release 鼠标的时候其值就为负数,以此来达到区分鼠标左右键同时点击与鼠标单击事件的效果。

**status:** Status 枚举型对象,用来记录当前方块的状态。

### Label 函数成员:

**Label(Status, int row, int col):** 调用 **this(s, new Point(x, y))**。

**Label(Status, Point):** 设置 Label 初始位置及状态。

**setStatus(Status):** 改变 Label 状态,并使其显示对应的图片。

### Panel 静态常量:

**DISX:** 图片  [CLOCK.png] 与  [LABEL.png] 之间, 

[LABEL.png] 与  [MINE.png] 之间的距离。

**DISY:** 方块矩阵与 CLOCK.png 等图片之间的距离。

**X:** 方块矩阵在 Demine 默认 JPanel 中的水平坐标。

**Y:** 方块矩阵在 Demine 默认 JPanel 中的垂直坐标。

### Panel 数据成员:

**col:** 方块矩阵的列数。

**row:** 方块矩阵的行数。

**mines:** 地雷的数量。

**frame:** 将主 Demine 传进来,用于关联游戏胜利时的弹出窗口。

**game:** 用于放置每一个 Label 的 JPanel 对象,其布局管理器为 GridLayout。


**label[row][col]:** Label 类的对象,用来表示 Label 的二维数组,行和列的值由数据成员 row 和 col 决定。

**mark[row][col]:** int 型数组,在内存中表示二维数组中该点是否为雷或者数字或者是空白块,与 label 二维数组之间由 Status 类的 getValue()、getStatus(int)方法关联。

**minecnt:** 记录剩下的旗子数 JLabel 对象,与 JLabel 的图片 



[LABEL.png]之间用 JLayeredPane 层次显示，由 Label 的监听器负责修改，当旗子数为 0 时，不能再放置旗子，当旗子被右击转换

为  时，旗子的数量增加 1。

**open:** 用于记录被点开的格子数，由方法 ClickLabel()负责修改，当某个 Label 被点开或者被宽度优先搜索到时，改值加一，由于扫雷不一定要点开全部的雷，而是点开所有非雷的方块即获胜，所有 open 值即为扫雷游戏获胜条件的判定，当  $open == row * col - mines$  时，游戏获胜。

**play:** 用于记录游戏状态，0 表示还未开始，这时候雷还没有撒下去，对任何点的双击都会变成点开。1 表示正在玩游戏，-1 表示游戏结束，游戏结束时点击任何一个 Label 都会重新开始游戏，由每一个 Label 的监听器负责修改。

**timecal:** 由 Timer 类对象 timer 负责修改，当游戏结束时停止计时，重置游戏计时为 0，开始游戏后开始计时。

**timer:** Timer 类对象，由 Label 的监听器负责修改。restart()与 stop()函数的调用时机与状态与 timecal 相同。

## Panel 函数成员:

**Panel(Demine f, int r, int c, int m):** 将主 Demine 传给 frame 引用，设置方块矩阵的行 r 和列 c，以及雷的数量 m。

**Panel(Demine f, int r, int c, int m, Label[][] l, int[][] mar):** 前面四个参数同上，最后两个二维数组是最初设计时想实现 win7 扫雷中保存游戏的功能，从文件读入上一次运行结束的局面，重新打开时能够读入，最后由于时间原因，没有实现其他部分的函数，导致保存游戏的功能没有实现。

**reSize(int r, int c, int m):** 在选择其他难度扫雷时，调用该方法来清除 Panel 中所有的原组件，并重绘一个行为 r，列为 c，雷为 m（雷无法重绘，只能设置一下新游戏中雷的数目），带有计时器和旗子数标记的 JLabel。

**addComponent():** 添加    组件及其上的数字显示，

数字与  [LABEL.png]用 JlayeredPane 布局管理器显示。

**clickLabel(Point p):** 点开位于 p 位置的 Label，若为雷，则游戏结束并显示所有雷，若  $mark[p.x][p.y]$  的值为 0-8，则进行宽度优先搜索翻开搜索到的 Label，并负责判断翻开后是否达到游戏胜利条件（open 值的改变由该函数负责）。

**Init():** 重置旗子数、时间以及所有 Label 的状态，该函数与 reSize()重绘面板函数不冲突，实际上 reSize()函数在调用了 addComponent()函数后又调用了该函数，该函数也在游戏重新开始时被调用。

**Srand(Point p):** 随机撒雷，保证位置为 p 的 Label 不被撒到雷。

## Demine 数据成员:

**level:** 用于记录游戏的三个等级，由易到难分别为 0, 1, 2

**panel:** 是内部类 Panel 的对象，Panel 类负责的内容如上。

**rank:** 是内部类 Rank 的对象，Rank 类负责排行榜。

### Demine 函数成员:

**Demine():** 负责运行程序时，添加的菜单及菜单监听器、排行榜读入文件及各个数据成员的初始化，程序结束时的排行榜写入文件等工作。

**ReSize():** 选择新的难度时，重绘 panel 并重设 JFrame 大小。

## 2. 各功能模块介绍

游戏运行时的各个模块功能如上，这里主要介绍菜单各项的功能。

**新游戏:** 无论处在什么样的游戏状态，丢弃上一局的游戏数据，强制开始新的一盘游戏，难度等级与上一局相同。

**统计:** 弹出排行榜对话框。

**选项:** 弹出难度选择对话框，必须关闭该对话框才能点击主 Frame。

**退出:** 将排行榜写入文件并退出。

**帮助:** 打开网页[帮助](#)。

**关于扫雷:** 无。

**联机获取更多游戏:** 打开网页[联机获取更多游戏](#)。

## 3. 系统运行界面

